08-07-00

A

PATENT APPLICATION
Attorney's Do. No. 5038-55
Intel #P9222

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

EXPRESS MAIL	MAILING LABEL NO. EL432977368US DATE OF DEPOSIT: AUGUST 2, 2000

I HEREBY CERTIFY THAT THIS PAPER AND ENCLOSURES AND/OR FEE ARE BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 CFR 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO: BOX PATENT APPLICATION, ASSISTANT COMMISSIONER FOR PATENTS, WASHINGTON D.C. 20231.

Melissa C. Smith — (SENDER'S PRINTED NAME)

Melissa C. Smith — (SIGNATURE)

Box Patent Application
Assistant Commissioner for Patents
Washington, D.C. 20231

Enclosed for filing is a patent application under 37 CFR 1.53(b) of:

Inventors: Rajesh R. Shah
For: DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A CHILD DRIVER

[If continuing application] This application is a [ ] continuation, [ ] divisional, [ ] continuation-in-part of prior application Serial No. _____, filed _____.

Enclosures:
[X] Specification (pages 1-6); claims (pages 7-10); abstract (page 11)
[X] 10 sheets of drawings
[X] Declaration or Combined Declaration and Power of Attorney
- [X] Newly executed (original or copy)
- [ ] Copy from a prior application (37 CFR 1.63(d))
- [ ] Incorporation by Reference--The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
- [ ] Deletion of Inventors (signed statement attached deleting inventor(s) named in the prior application (37 CFR 1.63(d)(2) and 1.33(b)

[ ] Power of Attorney

[X] Assignment with cover sheet
[ ] Certified copy of priority document:
[ ] Information Disclosure Statement with Form PTO 1449
[ ] Copies of references listed on attached Form PTO-1449
[ ] Preliminary Amendment
[ ] Change of Address

CLAIMS AS FILED				
For	Number Filed	Number Extra	Rate	Basic Fee $690.00
Total Claims	27-20	7	x $ 18 =	126.00
Independent Claims	3-3	0	x $ 78 =	0.00
Multiple Dependent Claim Fee			x $260 =	0.00
TOTAL FILING FEE				$ 816.00

[ ] Cancel in this divisional application original claims _____ of the prior application Serial No. _____ before calculating the filing fee. (At least one original independent claim must be retained for filing purposes.)

[X] A check in the amount of $856.00 to cover [X] filing fee and [X] assignment recordal fee ($40) is enclosed.

[X] Any deficiency or overpayment should be charged or credited to deposit account number 13-1703. A duplicate copy of this sheet is enclosed.

Customer No. 20575

Respectfully submitted,

MARGER JOHNSON
& McCOLLOM, P.C.

Ariel Rogson
Registration No. 43,054

MARGER JOHNSON
& McCOLLOM, P.C.
1030 S.W. Morrison Street
Portland, OR 97205
(503) 222-3613

Docket No. P9222

PATENT

UNITED STATES APPLICATION FOR LETTERS PATENT

for

DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A CHILD DRIVER

By

Rajesh R. Shah

Filed

August 2, 2000

DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A CHILD DRIVER

FIELD OF THE INVENTION

This invention pertains to computer operating systems, and more particularly to dynamic removal of driver stacks from the operating system.

BACKGROUND OF THE INVENTION

Although computer systems begin with the hardware components, computer operation requires software that controls the use of the hardware components. Typically handled by the operating system running on the computer, these software elements are called *device drivers* (sometimes shortened to *drivers*). The device drivers are typically specific to the hardware component they support. Because accessing a particular hardware component requires cooperation among multiple software drivers, those device drivers are grouped into *driver stacks* (sometimes shortened to *stacks*). The drivers in each stack work together, allowing access to the desired hardware component.

FIG. 1 shows software hierarchy 105 of software drivers to access a hard disk. Software hierarchy 105 is typical of hierarchies for accessing a particular hardware component. At the top of hierarchy 105 is the Peripheral Component Interconnect (PCI) bus driver. Below that is the Small Computer System Interface (SCSI) adapter driver. The SCSI adapter includes a SCSI port, onto which is connected the hard disk, which has a disk partition on which data may be stored. For each of the devices in the hierarchy, device drivers are needed to control access to the hardware devices. Below the SCSI adapter driver is the SCSI port driver.

Each device driver may export services specific to that device driver. These services may be used by other drivers in the driver stack or by other programs, such as the operating system. FIG. 2A shows driver stack 205 with three device drivers 210, 215, and 220, each device driver providing a service. Driver 215 is using the service provided by driver 210 (as shown by arrow 225), and driver 220 is using the service provided by driver 215 (as shown by arrow 230).

Each driver needs to know if other drivers are using its services. This information is important in case the driver stack is to be removed (see below). Because their services are being used, drivers 210 and 215 have non-zero reference counts. Note that it is not important for the driver to know who is using its service, only that its service is being used.

Although FIG. 2A shows only children drivers using the services of their immediate parents, this is not a limitation of driver services export. For example, in FIG. 2B, driver 210 in driver stack 205 is shown using the services exported by its "grandchild," driver 220 (shown by arrow 235).

It may happen that a driver stack, loaded to allow access to a particular hardware component, is no longer required. For example, the hardware component in question may have been removed from the computer system. When a driver stack can be removed from the operating system while the computer is in use, the driver stack is called *dynamic*. The operating system interrogates each driver in the stack, asking the drivers if they may be removed. Then, if each driver in the stack approves the remove query, the driver stack is removed. (Static driver stacks are also possible, but in a static driver stack, the driver stack may not be removed while the computer is running. The static driver stack can only be "removed" by specifying it not be loaded when the computer is next started.)

FIGs. 3A and 3B show the procedure used to query whether a driver stack may be removed. At block 305, the lowest device in the driver in the stack receives the remove query. At decision point 310, the device checks to see if its services are being used by any other programs. If its services are being used, then at block 315 the device signals the operating system that the driver stack may not be removed. Otherwise, at decision point 320 the device checks to see if it is using any services provided by other devices. If it is, then at block 325 the device stops using the service, and at block 330 the device decrements the reference count of the devices whose services it was using. At decision point 335, the device checks to see if it has a parent device in the driver stack. If it does, then at block 340 the device passes the remove query to its parent device, and the process returns to block 305. Otherwise, the last device in the driver stack has approved the remove query, and at block 345 the device signals the operating system that the driver stack may be removed.

A person skilled in the art will recognize that the procedure of FIGs. 3A and 3B will fail when a parent device uses the services of a child device, as shown in FIG. 2B, because the child device must approve the remove query before the parent device receives notice of the remove query. Referring to FIG. 2B, child driver 220 will note that its services are being

used and immediately fail the remove query. Parent driver 210 never has a chance to stop using the services of child driver 220, allowing the remove query to succeed.

The present invention addresses these and other problems associated with the prior art.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a hierarchy of device driver object components in a computer system requiring a driver stack.

FIG. 2A shows a typical driver stack.

FIG. 2B shows the driver stack of FIG. 2A with a parent driver accessing a service of a child driver.

FIGs. 3A and 3B show the procedure used by an operating system to remove a dynamic driver stack.

FIG. 4 shows a computer system running an operating system that may dynamically remove driver stacks in accordance with the invention.

FIG. 5 shows the driver stack of FIG. 2B using a virtual device object in accordance with the invention.

FIG. 6 shows a virtual device object in accordance with the invention redirecting a remove query to the parent driver in the operating system of FIG. 4.

FIG. 7 shows the procedure used to enable a parent device to access a service provided by a child device in the operating system of FIG. 4 in accordance with the invention.

FIG. 8 shows the procedure used by the virtual device to process a remove query received from the operating system of FIG. 4 in accordance with the invention.

DETAILED DESCRIPTION

FIG. 4 shows a computer system 405 in accordance with the invention. Computer system 405 includes a computer 410, a monitor 415, a keyboard 420, and a mouse 425. Computer 410 includes hardware components, such as a central processing unit, a memory, and a cache (not shown). Computer system 405 may also include other equipment not shown in FIG. 4, for example, other input/output equipment or a printer.

Running on computer system 405 is operating system 430. Operating system 430 includes drivers, such as drivers 435-1, 435-2, 435-3, 435-4, and 435-5. These drivers are

organized into driver stacks, such as 440-1 and 440-2. Note that driver stacks 440-1 and 440-2 are for exemplary purposes only. There may be more than two driver stacks loaded in operating system 430, and driver stacks 440-1 and 440-2 may have differing numbers of drivers within each driver stack. A person skilled in the art will also recognize that drivers may be duplicated in multiple driver stacks: for example, driver 435-2 is included in both driver stacks 440-1 and 440-2.

FIG. 5 shows how driver stack 205 of FIG. 2B may be implemented according to the invention to allow for the removal of driver stack 205. In FIG. 5, parent driver 210 creates virtual device object 505. Virtual device object 505 is inserted into driver stack 205 below the accessed child device 220. In practice, virtual device object 505 will be placed at the bottom of driver stack 205. But if the implementation allows, virtual device object 505 may be inserted anywhere in driver stack 205, provided virtual device object 505 is below the accessed child device 220. Virtual device object 505 then accesses the service of child device 220 on behalf of parent device 210, as shown by dashed line 510. Virtual device object 505 is "bound" to parent device 210, as shown by line 515. In effect, virtual device object 505 is a "placeholder" for parent device 210. Parent device 210 "fools" child device 220 into thinking child device 220 is being accessed by a lower child device.

By creating virtual device object 505 and placing it in the driver stack below child device 220 whose services are being accessed, the standard remove query procedure of FIGs. 3A and 3B may be used. The remove query will be delivered to virtual device object 505 before it is delivered to child device 220. Virtual device object 505 then informs parent device 210 of the remove query. Parent device 210 can stop using the service of child device 220. Virtual device 505 may then approve the remove query and pass the remove query to child device 220. Because parent device 210 is no longer using the services of child device 220, child device 220 may also approve the remove query. The remove query iterates up the driver stack, and ultimately may be approved by every driver in the driver stack. The operating system is then able to remove the driver stack.

A person skilled in the art will recognize that this technique may be extended beyond the case of a single parent device accessing services of a single child device. Each parent device that wants to access a service of a child device may add a virtual device object to the driver stack below the accessed child device. Further, a single parent device may access the services of multiple child devices using a single virtual device object, provided the virtual

device object is below all the child devices whose services are being used by the parent device.

It may happen that a new child device is added to the driver stack below the virtual device, the new child device providing a service desired by the parent device. The parent device may either add a new virtual device object or relocate the existing virtual device below the new child device in the driver stack. (In practice, it is preferable for the parent device to add a new virtual device object below the new child device and use the new virtual device object only for accessing the services of the new child device.)

Because the virtual device object is a placeholder for the parent device, the parent device is actually accessing the service of the child device. When the virtual device object receives a remove query from the operating system, the virtual device object lets the parent device know that the parent device should stop using the services of the child device. Referring to FIG. 6, since virtual device object X 505 is "bound" to driver A 210, when the operating system 430 sends remove query 605 to device object X 505 it is actually sending remove query 605 to driver A 210. Driver A 210 can then stop using the services of the child device, and virtual device 505 may pass remove query 605 to the next device driver in the driver stack.

There are several ways that virtual device 505 can inform parent device 210 about remove signal 605. One way is to use events, as described in object-oriented programming. Another way is for the operating system to directly invoke the code in the virtual device object. This code may directly link to code in the parent device for processing remove queries. A person skilled in the art will also recognize other techniques for passing the remove query from the virtual device object to the parent device.

FIG. 7 shows the procedure used to enable a parent device to access a service provided by a child device in the operating system of FIG. 4 in accordance with the invention. At block 705, the new virtual device object is created. At block 710, the virtual device object is bound to the parent device. At block 715, the virtual device object is inserted into the driver stack below the child device whose services are sought. At block 720, the parent increments the reference count of the child device. Finally, at block 725, the parent device accesses the services of the child device.

FIG. 8 shows the procedure used by the virtual device to process a remove query received from the operating system of FIG. 4 in accordance with the invention. At block 805, the virtual device receives the remove query. At block 810, the parent device stops using the

services of the child device. At block 815, the parent device decrements the reference count of the child device. Finally, at block 820, the virtual device object passes the remove query to the next device in the driver stack.

Having illustrated and described the principles of my invention in an embodiment thereof, it should be readily apparent to those skilled in the art that the invention can be modified in arrangement and detail without departing from such principles. I claim all modifications coming within the spirit and scope of the accompanying claims.

I claim:

1. A method for a parent device to access a service of a child device in a driver stack, the method comprising:

creating a virtual device;

binding the virtual device to the parent device;

inserting the virtual device in the driver stack below the child device; and

accessing the service of the child device.

2. A method according to claim 1, wherein the driver stack is a dynamic driver stack.

3. A method according to claim 2, wherein accessing the service of the child device includes accessing the service of the child device by the parent device.

4. A method according to claim 2, wherein accessing the service of the child device includes incrementing a reference count of a number of users of the service of the child device.

5. A method according to claim 2, wherein binding the virtual device includes arranging the parent device to receive a query to remove the dynamic driver stack sent to the virtual device.

6. A method according to claim 2, the method further comprising:

receiving at the virtual device a query to remove the dynamic driver stack;

releasing the service of the child device; and

passing the query to remove the dynamic driver stack to a next device in the dynamic driver stack.

7. A method according to claim 6, wherein releasing the service of the child device includes releasing the service of the child device by the parent device.

8. A method according to claim 6, wherein releasing the service of the child device includes invoking a code within the virtual device that accesses the parent device.

9. A method according to claim 6, wherein releasing the service of the child device includes decrementing a reference count of a number of users of the service of the child device.

10. A method according to claim 2, wherein accessing the service of the child device includes accessing a second service of a second child device above the virtual device in the dynamic driver stack.

11. An article comprising:

a storage medium, said storage medium having stored thereon instructions, that, when executed by a computing device, result in:

creating a virtual device;

binding the virtual device to the parent device;

inserting the virtual device in a driver stack below the child device; and

accessing the service of the child device.

12. An article according to claim 11, wherein the driver stack is a dynamic driver stack.

13. An article according to claim 12, wherein accessing the service of the child device includes accessing the service of the child device by the parent device.

14. An article according to claim 12, wherein accessing the service of the child device includes incrementing a reference count of a number of users of the service of the child device.

15. An article according to claim 12, wherein binding the virtual device includes arranging the parent device to receive a query to remove the dynamic driver stack sent to the virtual device.

16. An article according to claim 12, the storage medium having stored thereon further instructions that, when executed by the computing device, result in:

receiving at the virtual device a query to remove the dynamic driver stack;

releasing the service of the child device; and

passing the query to remove the dynamic driver stack to a next device in the dynamic driver stack.

17. An article according to claim 16, wherein releasing the service of the child device includes releasing the service of the child device by the parent device.

18. An article according to claim 16, wherein releasing the service of the child device includes invoking a code within the virtual device that accesses the parent device.

19. An article according to claim 16, wherein releasing the service of the child device includes decrementing a reference count of a number of users of the service of the child device.

20. An article according to claim 12, wherein accessing the service of the child device includes accessing a second service of a second child device above the virtual device in the dynamic driver stack.

21. An apparatus supporting removal of a driver stack, the apparatus comprising:

a computer including a hardware component requiring the driver stack;

an operating system running on the computer;

the driver stack loaded onto the operating system and supporting the hardware component, the driver stack including at least a parent driver and a child driver, the child driver providing a service accessed by the parent driver; and

a virtual driver installed below the child driver in the driver stack.

22. An apparatus according to claim 21, wherein the operating system is designed to support dynamic removal of the driver stack.

23. An apparatus according to claim 22, wherein the virtual driver is adapted to inform the parent driver when the driver stack is to be removed.

24. An apparatus according to claim 22, wherein the parent driver is adapted to insert the virtual driver into the driver stack before accessing the service provided by the child driver.

25. An apparatus according to claim 22, wherein the child driver includes a reference count of a number of users of the service.

26. An apparatus according to claim 26, wherein the parent driver is adapted to increment the reference count of the child driver before accessing the service provided by the child driver.

27. An apparatus according to claim 26, wherein the parent driver is adapted to decrement the reference count of the child driver after being informed by the virtual driver that the driver stack is to be removed and stopping use of the service provided by the child driver.

DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A CHILD DRIVER

ABSTRACT OF THE DISCLOSURE

A parent driver desiring to access a service of a child driver in a driver stack creates a virtual device object. The virtual device object is inserted into the driver stack below the child driver. When a query to dynamically remove the driver stack arrives at the virtual device object, the virtual device object notifies the parent driver. The parent driver stops accessing the child driver before directly receiving and processing the remove query, allowing the driver stack to be removed after all drivers in the driver stack process the remove query.

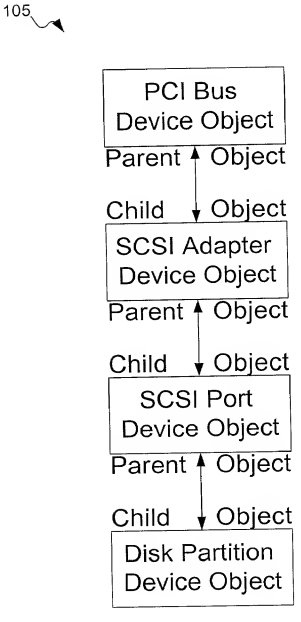

FIG. 1
(Prior Art)

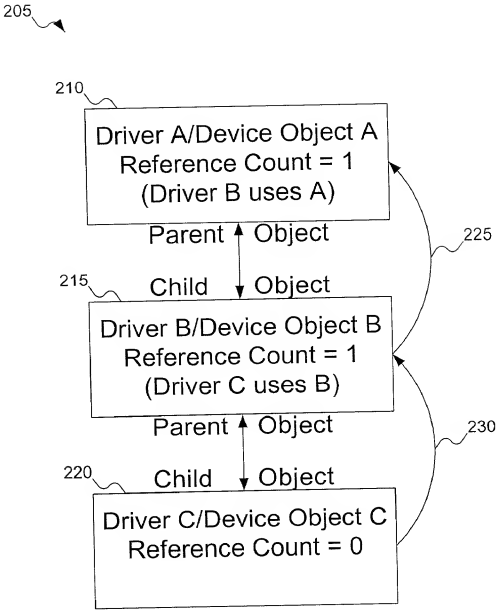

FIG. 2A
(Prior Art)

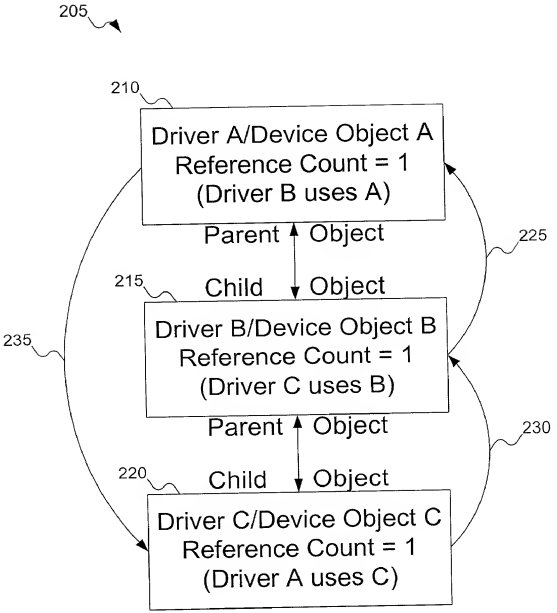

FIG. 2B
(Prior Art)

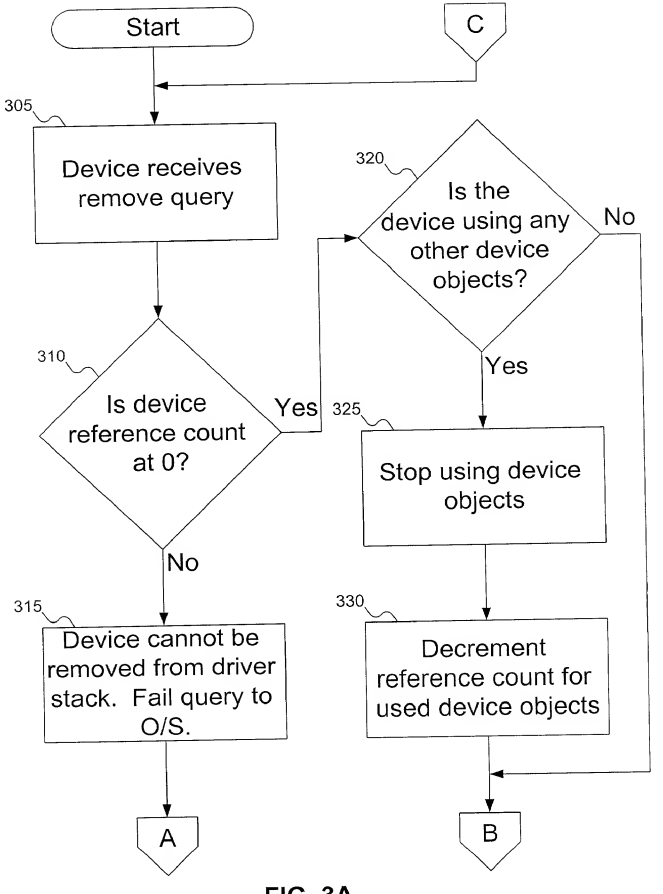

FIG. 3A
(Prior Art)

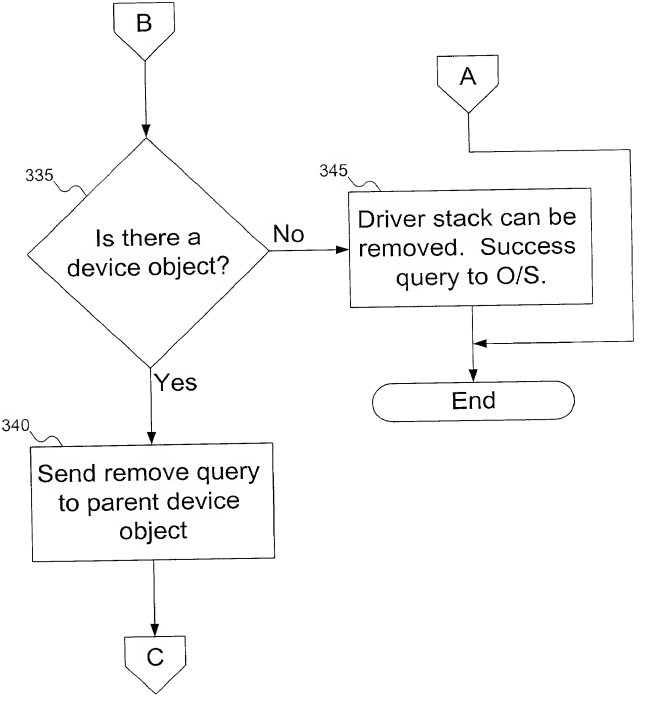

FIG. 3B
(Prior Art)

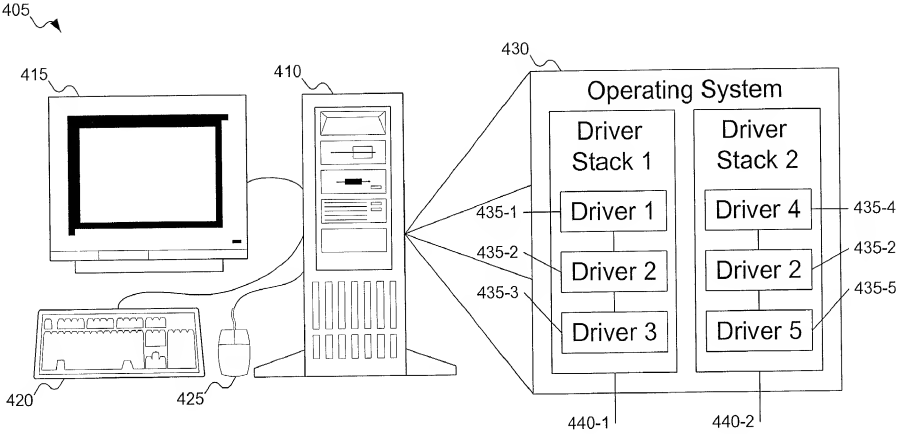
405
415
410
430
Operating System
Driver Stack 1
Driver Stack 2
435-1
Driver 1
435-2
Driver 2
435-3
Driver 3
Driver 4
435-4
Driver 2
435-2
Driver 5
435-5
420
425
440-1
440-2

FIG. 4

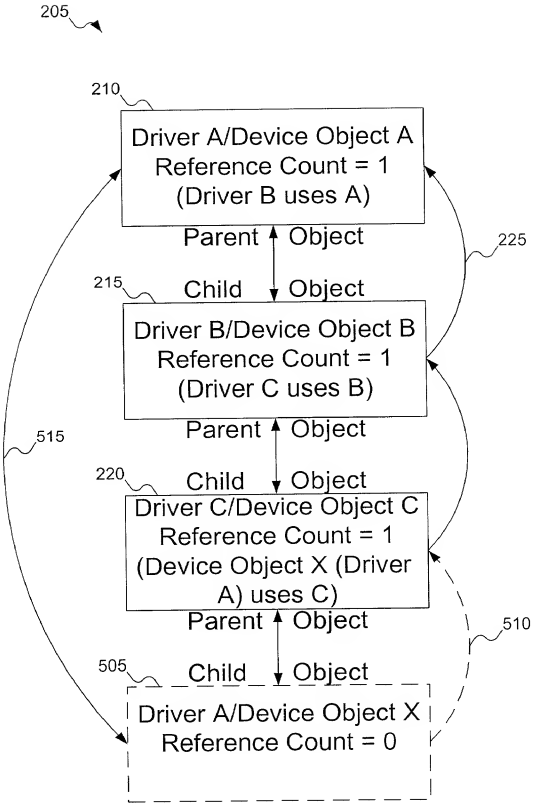
205
210
Driver A/Device Object A
Reference Count = 1
(Driver B uses A)
Parent Object
225
215
Child Object
Driver B/Device Object B
Reference Count = 1
(Driver C uses B)
Parent Object
515
220
Child Object
Driver C/Device Object C
Reference Count = 1
(Device Object X (Driver
A) uses C)
Parent Object
510
505
Child Object
Driver A/Device Object X
Reference Count = 0

FIG. 5

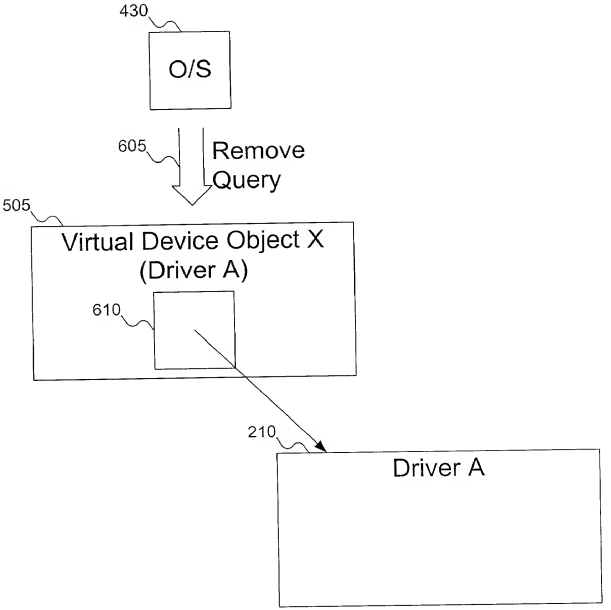
430
O/S
605
Remove
Query
505
Virtual Device Object X
(Driver A)
610
210
Driver A

FIG. 6

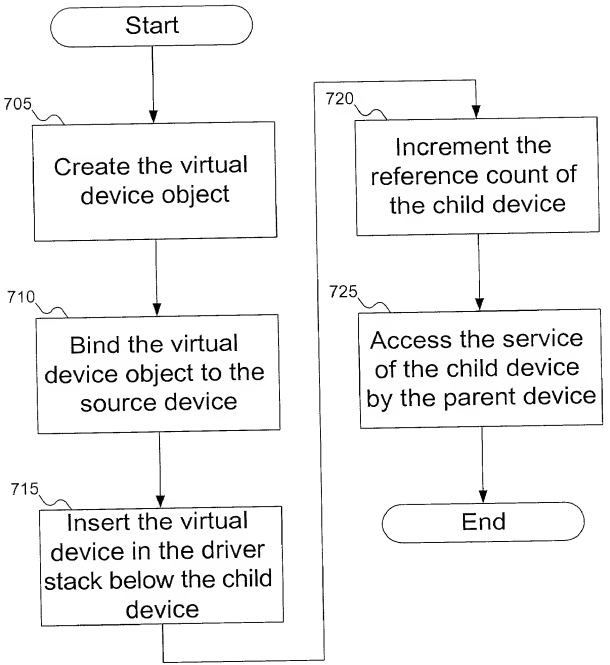
Start
705
Create the virtual device object
710
Bind the virtual device object to the source device
715
Insert the virtual device in the driver stack below the child device
720
Increment the reference count of the child device
725
Access the service of the child device by the parent device
End

FIG. 7

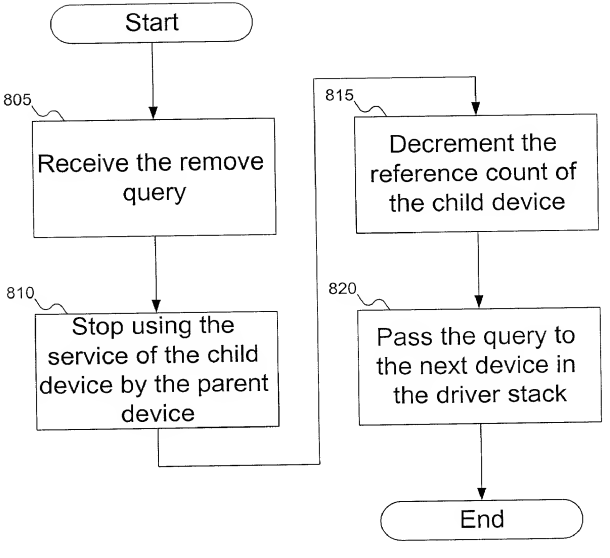
Start
805
Receive the remove query
810
Stop using the service of the child device by the parent device
815
Decrement the reference count of the child device
820
Pass the query to the next device in the driver stack
End

FIG. 8

PATENT APPLICATION
Attorney Docket No. 5038-55
Client Ref. No. P9222

COMBINED DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention which DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A CHILD DRIVER.

[X] is attached hereto.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, Sec. 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Sec. 119 (a)-(d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below any foreign application for patent or inventor's certificate, or of any PCT international application having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Claiming Priority?	
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[ ] No

I hereby claim the benefit under Title 35, United States Code, Sec. 119(e) of any United States provisional application listed below:

Provisional Application No.	Filing Date

I hereby claim the benefit under Title 35, United States Code, Sec. 120 or §365(c) of any PCT international application designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Sec. 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Sec. 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application No.) (Filing Date) (Status) (patented, pending, abandoned)

I hereby appoint the following attorneys to prosecute the application, to file a corresponding international application, to prosecute and transact all business in the Patent and Trademark Office connected therewith:

Customer No. 20575

Attorney Name	Registration No.
Jerome S. Marger	26,480
Alexander C. Johnson, Jr.	29,396
Alan T. McCollom	28,881
James G. Stewart	32,496
Glenn C. Brown	34,555
Stephen S. Ford	35,139
Gregory T. Kavounas	37,862
Scott A. Schaffer	38,610
Joseph S. Makuch	39,286
James E. Harris	40,013
Graciela G. Cowger	42,444
Ariel Rogson	43,054
Craig R. Rogers	43,888

Direct all telephone calls to Alan T. McCollom at (503) 222-3613 and send all correspondence to:

MARGER JOHNSON & McCOLLOM, P.C.
1030 S.W. Morrison Street
Portland, Oregon 97205

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Rajesh R. Shah

Inventor's signature: Rajesh R. Shah Aug. 1, 2000 (Date)

Residence: Portland, Oregon

Citizenship: India

Post Office address: 14320 N.W. Lilium Drive
Portland, Oregon 97229